MS-DOS. Команды, применяемые в bat-файлах.

ECHO [ON|OFF]
В bat-файле включает/выключает вывод текста команд, в командной строке - вывод промптера. Без параметра - выводит текущую установку ECHO. Команда ECHO может находиться в команде IF.

ECHO текст
Выводит текст. Команда **ECHO.** (с точкой) выводит пустую строку.

@ перед командой (в bat-файле)
Не выводить на экран текст самой команды.

PAUSE
Приостанавливает выполнение bat-файла до нажатия клавиши и выводит текст:
Press any key to continue . . .

TIMEOUT [/T] timeout [/NOBREAK]
Приостанавливает выполнение bat-файла на timeout секунд или до нажатия клавиши.
/T timeout Ключ /T имеет только декоративную роль. Допустимы значения от –1 до 9999. –1 означает бесконечное ожидание нажатия клавиши.
/NOBREAK Ожидает указанное время, игнорируя все нажатия клавиш, кроме сочетания Ctrl+C

REM [текст]
Комментарий. В тексте комментария недопустимы символы > < |

GOTO метка
Переход на метку (используется только в bat-файлах). Любая строка с двоеточием в первой позиции считается меткой. Метки не могут включать разделителей пробел, точка с запятой и знак равенства.

EXIT /B [ERRORLEVEL]
Выход из bat-файла, не выходя из окна.
ERRORLEVEL При выходе из bat –файла устанавливает заданный ERRORLEVEL.

SHIFT [/n]
Изменяет значения параметров-переменных bat-файла (от %0 до %9), копируя каждый параметр в предыдущий (%0:=%1; %1:=%2, ...). Используется для выполнения одних и тех же операторов для некоторого количества параметров. Работает и более чем с 10 параметрами командой строки (после SHIFT'а 11-й параметр командой строки попадает в %9).
/n Сдвиг начинается с %n. Например, SHIFT /2 сдвигает %3 в %2, %4 в %3 и т.д., не изменяя %0 и %1.

Дополнительно параметры-переменные bat-файла (%0, %1, и т.д.) могут подвергаться таким преобразованиям (всех примеры используют параметр %1 и предполагают, что он содержит имя файла, полное или частичное):

```
%~1       - убирает в %1 ограничивающие кавычки (")
%~f1      - заменяет %1 на полное имя (со всем путем)
%~d1      - заменяет %1 на только имя диска (с двоеточием)
%~p1      - заменяет %1 на только путь (без имени диска)
%~n1      - заменяет %1 на только имя файла
%~x1      - заменяет %1 на только расширение файла
%~s1      - заменяет в %1 все имена на короткие (8.3)
%~a1      - заменяет %1 на только атрибуты файла
%~t1      - заменяет %1 на только дату и время последнего изменения
%~z1      - заменяет %1 на только размер файла
%~$PATH:1 - ищет файл в каталогах из списка PATH и, найдя его
            (первый соответствующий каталог), заменяет %1 на полное
            имя. Если не находит, то заменяет %1 на пустую строку.
```

Модификаторы можно комбинировать:

```
%~dp1       - заменяет %1 на только имя диска и путь
%~nx1       - заменяет %1 на только имя и расширение
%~dp$PATH:1 – тот же поиск файла в списке PATH, но в случае успешного
        поиска заменяет на имя первого найденного файла и путь к нему
%~ftza1     - заменяет %1 на запись, подобную записи в команде DIR
```

Подстановка значения переменной в команду: для того, чтобы подставить значение переменной (а не её название) в выполняемый оператор, необходимо заключить имя переменной в знаки процента: **%variable%**. Кроме того, допускаются следующие возможности модификации значения переменной при подстановке её значения:

%variable:str1=str2% - заменяются все вхождения str1 на str2; возможно str2 - пустая

%variable:~offset[,length]% - берём только length символов, начиная с (offset+1)-го (первые offset символов пропускаются); если length не специфицируется, то берётся весь остаток строки до конца; если offset или length отрицательны, то к их значениям добавляется длина строки.

Примеры:

set lala=abcdefgh	**Результат:**
echo %lala%	**abcdefgh**
echo %lala:abcd=dcba%	**dcbaefgh**
echo %lala:~1,5%	**bcdef**
echo %lala:~3%	**defgh**
echo %lala:~-3%	**fgh**
echo %lala:~0,-2%	**abcdef**

Условный оператор **IF**. Существует в нескольких формах:

IF [NOT] ERRORLEVEL число команда

Условие выполняется, если ERRORLEVEL >= число

IF [NOT] строка1==строка2 команда

Условие выполняется при точном совпадении строк.

строка - это строка символов или переменная (например, %1); строки не выделяются апострофами или кавычками.

IF [NOT] EXIST имя_файла команда

Условие выполняется, если указанный файл существует; в имя_файла допускаются метасимволы * и ?. Также применима для того, чтобы проверить имеется ли указанный диск. Для этого следует вместо имя_файла написать диск:

Допускается применение **ELSE**, например:

```
IF EXIST name.ext (
    del name.ext
) ELSE (
    echo name.ext missing.
)
```

либо

```
IF EXIST name.ext (del name.ext) ELSE echo name.ext missing
```

Следующие два примера не работают:

```
IF EXIST name.ext del name.ext ELSE echo name.ext missing
```

- команды первой части должны выделяться скобками, даже если это одна команда

```
IF EXIST name.ext del name.ext
ELSE echo name.ext missing
```

- ELSE в таком случае рассматривается как отдельная команда (а такой команды нет)

IF [/I] строка1 оператор_сравнения строка2 команда

```
оператор сравнения:      EQU - =        NEQ - <>
                         LSS - <        LEQ - <=
                         GTR - >        GEQ - >=
```

/I при сравнении учитывать регистр букв (различать большие/маленькие)

Если строки представляют из себя записи чисел, то сравнение числовое.

Выражение %ERRORLEVEL% дает строковое представление ERRORLEVEL. Такое представление можно использовать с командами сравнения (EQU и т.д.) или в команде GOTO. Например:

```
goto answer%ERRORLEVEL%        IF %ERRORLEVEL% LEQ 1 goto okay
:answer0
...
:answer1
...
```

IF DEFINED variable command
Действует подобно IF EXIST, только проверяет наличие переменной среды.

IF CMDEXTVERSION number command
Действует подобно IF ERRORLEVEL, только проверяет версию ОС.

Операторы объединения команд: & && ||

command1 & command2
Просто две (или больше) команды выполняются одна за другой — эквивалентно тому, что команды записаны одна за другой каждая в своей строке.

command1 && command2
Оператор && гарантирует, что вторая команда будет выполнена только, если первая команды была выполнена успешно, т. е. с нулевым ERRORLEVEL.

command1 || command2
Такая запись означает, что вторая команда будет выполнена только тогда, когда первая команда завершилась неудачно (с ненулевым ERRORLEVEL). Например:

```
cd archives || exit 1
```

если перейти в каталог archives не удастся, то произойдёт выход с ERRORLEVEL 1

CALL [drive:][path]bat-файл [параметры bat-файла]
Вызывает один bat-файл из другого с возвратом обратно.
Рекурсия допустима, надо только не забывать о выходе.

параметры bat-файла – параметры, передаваемые в bat-файл в командной строке, например, ключи, имена файлов, какие-либо выражения, содержащие значения параметров от %0 до %9 либо переменных среды.

Допускается также рекурсивный вызов из bat-файла самого себя, причем выполнение вызова начинается с указанной метки:

CALL :метка [параметры bat-файла]

Но это именно рекурсивный вызов: для полного выхода надо будет постепенно выйти из всех выполняемых экземпляров bat-файла и даже при таком вызове (из самого себя) образуется новый полный комплект параметров %1, %2, ...
Такой синтаксис команды CALL позволяет организовать в bat-файле некотрое подобие вызова подпрограммы. Все переменные среды являются при этом вызове глобальными переменными, но параметры командной строки (от %1 до %9) – локальны в каждом вызове.

CHOICE [/C строка_букв] [/N] [/CS] [/T n /D буква] [/M текст]
Предлагает в bat-файле пользователю выбор: выводит указанный текст и приостанавливает работу до нажатия одной клавиши из списка в строке_букв. Затем устанавливает ERRORLEVEL по номеру позиции нажатой клавиши в списке строка_букв. Нажатие CTRL+C или CTRL+Break устанавливает ERRORLEVEL в 0. При неверном нажатии устанавливает ERRORLEVEL в 255.

/C строка_букв	Множество возможных клавиш-ответов. По умолчанию - YN.
/N	Не выводить строку_букв в промптере. Текст перед ней, тем не менее, выводится.
/CS	Различать большие и маленькие буквы. По умолчанию - не различаются.
/T n	Ожидать n (0..9999) секунд, после чего считается нажатой кнопка, указанная в /D
/M текст	Выводимый текст промптера; может быть заключен в кавычки

Примеры

```
CHOICE /C YNC /M "Press Y for Yes, N for No or C for Cancel."
CHOICE /T 10 /C ync /CS /D y
CHOICE /C ab /M "Select a for option 1 and b for option 2."
CHOICE /C ab /N /M "Select a for option 1 and b for option 2."
```

FOR %%variable IN (множество) DO команда [параметры] (в bat-файлах)
FOR %variable IN (множество) DO команда [параметры] (в командной строке)
Оператор цикла. Допустима только одна команда.

%%variable либо %variable Переменная – однобуквенный заменяемый параметр. Команда FOR подставляет вместо %%variable (%variable) строки текста из указанного множества и для них выполняется соответствующая команда. variable - это любая одна буква или цифра (%% в bat-файле необходимы во избежание конфликта с параметрами командной строки), большие и маленькие буквы в именах параметров различаются. Если в переменной содержится имя файла, то с ним допустимы такие же преобразования (%%~k, %%~fk, %%~dk и т.д.), как и с переменными параметрами (%0, %1, %2, ...)

множество Одна или несколько строк (или имен файлов). Скобки обязательно нужны. В именах файлов допустимы метасимволы * и ?.

Примеры.

`for %f in (*.doc *.txt) do type %f` Выводит все doc и txt файлы на экран.

`for %f in (*.doc *.txt) do type %f > prn:` Выводит все doc и txt файлы на принтер.

FOR /D %variable IN (множество) DO команда [параметры]
%variable применяется к каталогам, а не к файлам, в том числе и если в описании множества используются символы * и ?. Ключ /D можно совмещать с ключом /R

FOR /R [[диск:]путь] %variable IN (множество) DO команда [параметры]
Обходит все каталоги и подкаталоги, начиная с каталога, указанного в [диск:[путь]], и для каждого каталога исполняет всю команду FOR. По умолчанию начинает с текущего каталога. Если вместо множества указать `(.)`, то просто обходит все каталоги с пустым значением %variable.

FOR /L %variable IN (start,step,end) DO команда [параметры]
variable – последовательно принимает числовые значения от start до stop с шагом step.
Например, **`FOR /L %a IN (5,-1,0) DO echo %a`** выводит 5, 4, 3, 2, 1, 0

FOR /F ["options"] %variable IN (file-set) DO команда [параметры]
FOR /F ["options"] %variable IN ("string") DO команда [параметры]
FOR /F ["options"] %variable IN ('command') DO команда [параметры]
или, если используется опция usebackq:
FOR /F ["options"] %variable IN (file-set) DO команда [параметры]
FOR /F ["options"] %variable IN ('string') DO команда [параметры]
FOR /F ["options"] %variable IN (`command`) DO команда [параметры]

file-set – это одно или несколько имен файлов, в именах файлов недопустимы метасимволы * и ?. Каждый из файлов открывается, читается и обрабатывается, а затем происходит переход к следующему файлу. Обработка состоит в чтении файла, разбивании его на отдельные строки и разборе (parsing) каждой строки на один или несколько (возможно и 0) токенов. Затем вызывается тело цикла со строковыми переменными, которым присвоены значения найденных токенов. По умолчанию /F передает первый токен, отделенный пробелом, от каждой строки файла. Пустые строки пропускаются. Перекрыть поведение разбора по умолчанию можно, указав параметр ”options”.
Параметр ”options” (в кавычках) содержит одно или несколько ключевых слов, определяющих различные опции процесса разбора. Предусмотрены следующие ключевые слова:

eol=h одиночный символ – признак конца строки; текст вслед за первым появлением символа h игнорируется при разборе

skip=n первые n строк файла пропускаются

delims=xxx множество разделителей - заменяет разделители по умолчанию (пробел и табуляция).

tokens=x,y,m-n описывает, какие токены из каждой строки передаются в тело цикла при каждом его выполнении; это может привести к определению дополнительных переменных. m-n - это диапазон от m-го до n-го токена. Если последний символ строки **`tokens=`** - звездочка, то заводятся дополнительные переменные, которые получают значения из части строки, идущей вслед за частью, разобранной на токены.

usebackq указывает на использование новой записи: строка, ограниченная обратными апострофами (`` - акуты) выполняется как команда, а обычные апострофы (’ ’) ограничивают строку – это позволяет использовать двойные апострофы (” ” – кавычки) для ограничения имен файлов в file-set.

Пример. Команда

```
FOR /F "eol=; tokens=2,3* delims=, " %i in (myfile.txt) do @echo %i %j %k
```

разбирает каждую строку в myfile.txt, пропуская строки, начинающиеся с точки с запятой. Токены – это отрезки строки, разделенные пробелами и/или запятыми. В тело цикла передаются 2-й и 3-й токены из каждой строки. Заметим, что %i получает значение 2-го токена, %j - 3-го токена, а %k получает все оставшиеся токены после третьего. Если бы имена файлов содержали пробелы, то их следовало бы заключать в двойные кавычки. Чтобы использовать двойные кавычки таким способом, необходимо использовать опцию usebackq (иначе двойные кавычки будут протрактованы, как ограничители строки для разбора).

В данном примере %i явно объявлена в команде for, а %j и %k объявлены неявно через опцию tokens= . Таким способом возможно объявить до 26 токенов, но не допуская, чтобы объявлялись переменные с именами больше чем %z или %Z. Напомним, что имена переменных могут содержать ровно одну букву, причем большие и маленькие буквы различаются.

Логику разбора FOR /F можно применять и непосредственно к строкам (а не к построчному разбору файлов), используя соответствующие ограничители строки, записанной в ( ).

В конце концов, можно использовать команду FOR /F и для разбора вывода команд, ограничивая строку в ( ) соответствующим образом. Эта обработка происходит так: команда передается дочерней CMD.EXE, и ее выход размещается в памяти, а затем разбирается так, как будто это файл.

Примеры: `FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i`

выводит названия всех переменных среды (environment variables);

`FOR /F "tokens=*" %i IN ('cmd.exe /c ver') DO @echo %i`

выводит версию операционной системы (что–то вроде Microsoft Windows [Version 6.1.7601])

Кроме того переменная %variable может повергаться тем же преобразованиям, что и параметры переменные bat- файла (см. команду SHIFT): %~I, %~fI, %~dI, %~pI, %~nI, %~xI, %~sI, %~aI, %~tI, %~zI, %~$PATH:I

SETLOCAL

Включает локализацию изменений переменных среды в bat-файле. Все изменения переменных среды носят локальный характер и не выходят за пределы этого bat-файла – если при окончании работы bat-файла остаются незакрытые команды SETLOCAL, то для всех них автоматически выполняется команда ENDLOCAL

ENDLOCAL

Восстанавливает значения переменных среды на момент начала локализации.

Если включены расширения команд (Command Extensions), то **SETLOCAL** допускает следующие (опциональные) аргументы:

ENABLEEXTENSIONS/DISABLEEXTENSIONS

Включение/выключение расширений команд командного процессора. Эти аргументы имеют более высокий приоритет, чем ключи CMD /E:ON или /E:OFF

Если расширения команд включены, то появляется несколько динамически изменяемых переменных (они не выводятся командой SET, но вычисляются каждый раз при обращении к ним):

%CD% - текущий каталог.

%DATE% - текущая дата в том же формате, что и в команде DATE.

%TIME% - текущее время в том же формате, что и в команде TIME.

%RANDOM% - случайное число в диапазоне от 0 до 32767.

%ERRORLEVEL% - текущее значение ERRORLEVEL.

%CMDEXTVERSION% - расширенный номер версии действующего командного процессора

%CMDCMDLINE% - первоначальная командная строка, вызвавшая командный процессор, например: запуск bat-файла `D:\WorkDir\a.bat>set a`, состоящего из одной строки

```
set a = %cmdcmdline%
```

устанавливает значение переменной a

```
a = "C:\Windows\system32\cmd.exe" /C "D:\WorkDir\a.bat"
```

Если пользователь объявит переменную с таким же именем, то соответствующая динамически изменяемая переменная становится недоступной.

ENABLEDELAYEDEXPANSION/DISABLEDELAYEDEXPANSION

Включение/выключение **задержанной подстановки значений переменных**. Эти аргументы имеют более высокий приоритет, чем ключи CMD /V:ON или /V:OFF

При выполнении блока команд (например в командах if или for) подстановка значений переменных происходит до того, как они будет изменена в процессе выполнения, даже если переменные в блоке изменяются командой set. Иначе говоря, все переменные в блоке подставляют то свое значение, которое было у них на момент входа в блок. Для того, чтобы переменная подставляла свои значения, измененные внутри блока, необходимо, кроме установки

SETLOCAL ENABLEDELAYEDEXPANSION,

ограничивать переменную восклицательными знаками вместо знаков процента: !variable! вместо %variable%. Например,

```
set VAR=before
if "%VAR%" == "before" (
    set VAR=after
    if "%VAR%" == "after" @echo If you see this, it worked
)
```

не выведет сообщения. Чтобы сообщение всё-таки появлялось, следует включить отложенную подстановку значений переменных и заменить %VAR% внутри блока на !VAR!

```
setlocal ENABLEDELAYEDEXPANSION
set VAR=before
if "%VAR%" == "before" (
     set VAR=after
     if "!VAR!" == "after" @echo If you see this, it worked
)
```

Аналогично,

```
set LIST=
for %%i in (*.*) do set LIST=%LIST% %%i;
echo %LIST%
```

выведет только имя последнего файла (с точкой с запятой). Чтобы исправить ситуацию и вывести строку со списком всех файлов каталога, надо сделать так:

```
setlocal ENABLEDELAYEDEXPANSION
set LIST=
for %%i in (*.*) do set LIST=!LIST! %%i;
echo %LIST%
```

Обе модификации команды SETLOCAL действуют до соответствующей команды ENDLOCAL. Команда SETLOCAL с аргументом устанавливает ERRORLEVEL: если указано одно из двух возможных значений аргумента, то 0, в противном случае – 1.

Пример: bat-файл, проверяющий, включены ли расширения команд (Command Extensions):

```
@echo off
verify LALALA 2>nul
setlocal ENABLEEXTENSIONS 2>nul
if not ERRORLEVEL 1 (
  echo Extensions are enabled.
) else (
  echo Unable to enable extensions.
)
```

Команда `verify` (с некорректным параметром) нужна здесь именно для того, чтобы установить ERRORLEVEL в не 0 – в старых версиях CMD.EXE команда SETLOCAL не изменяет значения ERRORLEVEL.